CAPSTONE PROJECT PGP DSBA
SUPPLY CHAIN ANALYTICS

By Giridharan Velmurugan

Contents

List of Images and plots

List of Tables

Given Business Problem –

A FMCG company has entered into the instant noodles business two years back. Their higher management has notices that there is a miss match in the demand and supply. Where the demand is high, supply is pretty low and where the demand is low, supply is pretty high. In both the ways it is an inventory cost loss to the company; hence, the higher management wants to optimize the supply quantity in each and every warehouse in entire country.

Given Goal & Objective –

The objective of this exercise is to build a model, using historical data that will determine an optimum weight of the product to be shipped each time to the warehouse. Also try to analysis the demand pattern in different pockets of the country so management can drive the advertisement campaign particular in those pockets.

Problem Statement –

A FMCG company has identified a significant misalignment between supply and demand across different regions for their instant noodle product line.

Despite being in the market for two years, the company faces challenges with excess inventory in low-demand areas and shortages in high-demand areas, leading to inventory cost losses.

Need of the study –

To optimize inventory distribution to ensure that supply aligns more closely with regional demand patterns.

This optimization will reduce inventory costs, improve customer satisfaction by ensuring product availability, and enhance the company's market position.

Business Opportunity –

Develop a model to predict the optimum quantity of product shipments to each warehouse based on historical sales data.

Additionally, analysing demand patterns offers insights for targeted advertising campaigns, potentially increasing sales in high-demand areas.

Data Overview –

The dataset does not explicitly detail the time, frequency, and methods used for data gathering. However, certain attributes like “wh_est_year”, which provides the establishment year of warehouses, imply that the data spans several years. Fields such as “num_refill_req_l3m”, indicating the number of refill requests in the last three months, suggest that some data points are collected quarterly or done with time periods in mind. The dataset appears to be a mix of automatically collected data like refill requests and supply issues, and manually entered information such as warehouse establishment years and government certifications. This structured collection method encompasses various dimensions including warehouse capacity, location, and operational metrics, highlighting a systematic approach to data acquisition.

Numerical columns like “num_refill_req_l3m”, “transport_issue_l1y”, “Competitor_in_mkt” offer quantitative insights into the operational aspects of warehouse management and the competitive market landscape. Categorical data, including “Location_type”, “WH_capacity_size”, and “zone”, provide crucial segmentation of warehouses based on geographical location and capacity. The good thing is there are no duplicated full row instances, and the total dataset size is “586221” cell instances.

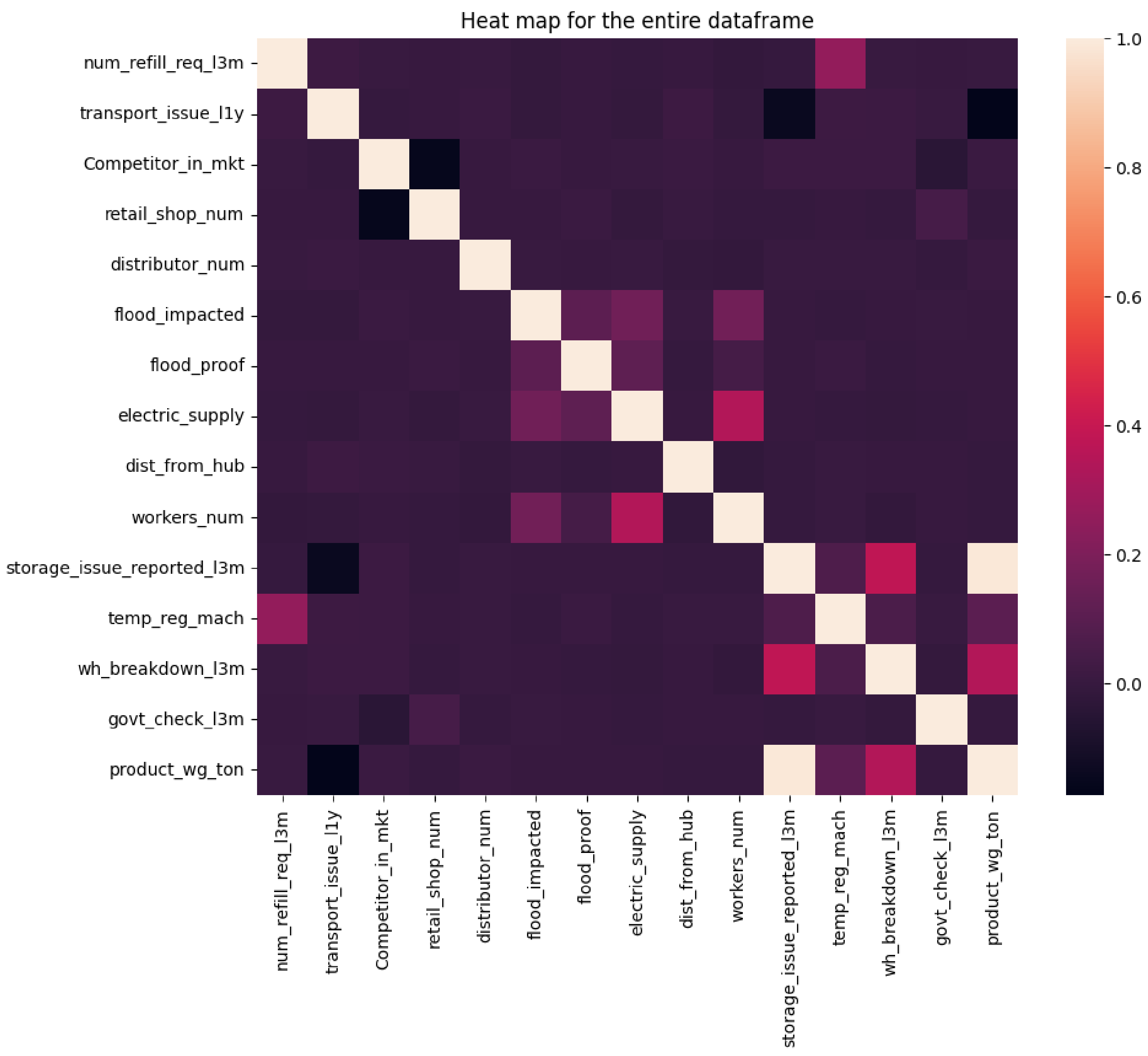
Understanding the dataset's attributes, particularly in the context of the FMCG company's goal to optimize supply and analyze demand patterns, highlights several key variables. “num_refill_req_l3m” serves as an indicator of demand, essential for supply optimization efforts. “product_wg_ton” (which is our target feature) reflects the supply quantity, which is directly linked to the company's objective of minimizing inventory costs through supply optimization. Additional factors like “Competitor_in_mkt”, “WH_capacity_size”, “electric_supply”, “flood_impacted” and others can influence demand patterns and are critical for developing targeted advertising campaigns.

Exploratory Data Analysis –

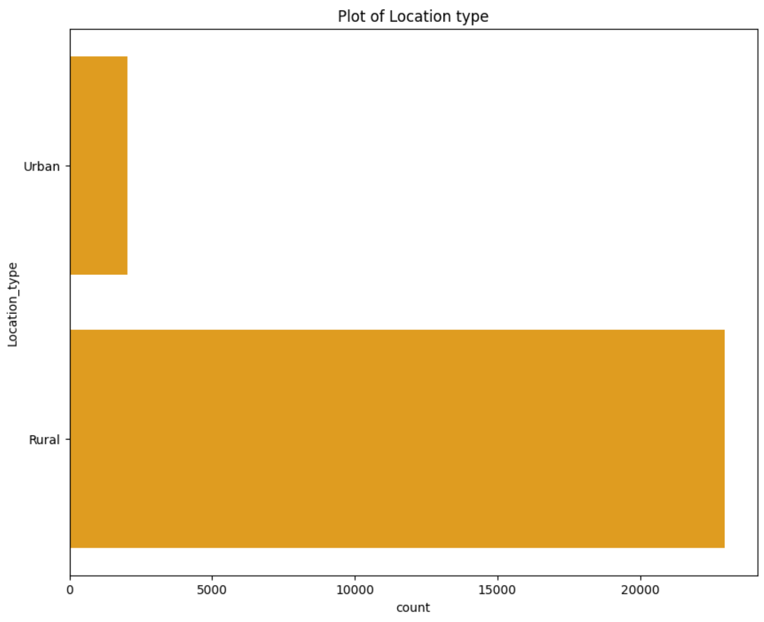
Initially we will be removing features "wh_est_year", "Ware_house_ID", "WH_Manager_ID". The former is removed because it has more than 30% of its cell instances as nulls. We are removing "Ware_house_ID" and "WH_Manager_ID" as they are just identifier features that do not convey any relevant information for our problem and in terms of model building.

Now let's dive in, below is a heat map of the features after above step. From the heat map we can see that there is very minimal to moderate and no correlation between the features, the target feature "product_wg_ton" is also present. This is not surprising as we have some categorical features.

The outlier in this space is feature "storage_issue_reported_l3m" which is strongly correlated to our target feature. This is a good thing as the influence will be helpful in model building and determining other factors.

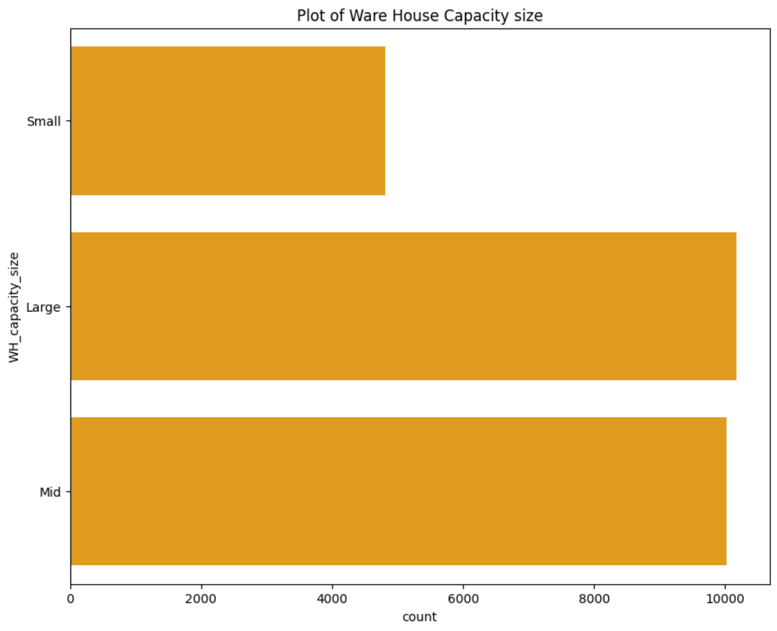


Location type – Below image shows the location type which is a categorical feature of two values "Urban" and "Rural". The dataset is unfortunately heavily sided towards rural environment. Out of 25000 full row instances, from the plot we can see more than 90% is of rural only. This will hinder while we are creating models as without enough examples it is difficult for the model to grasp the workings around this.

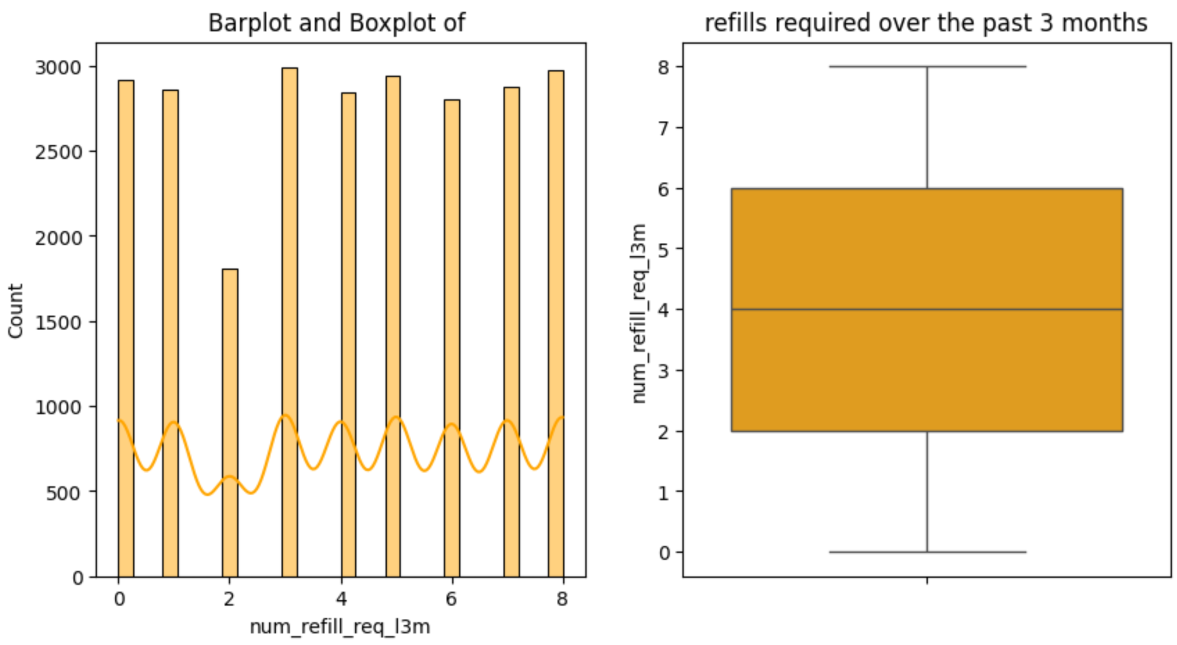


Warehouse capacity size – This is also a categorial feature of three values, "Small", "Mid" and "Large". Unlike location type the spread is comparatively better, but not what is typically expected.

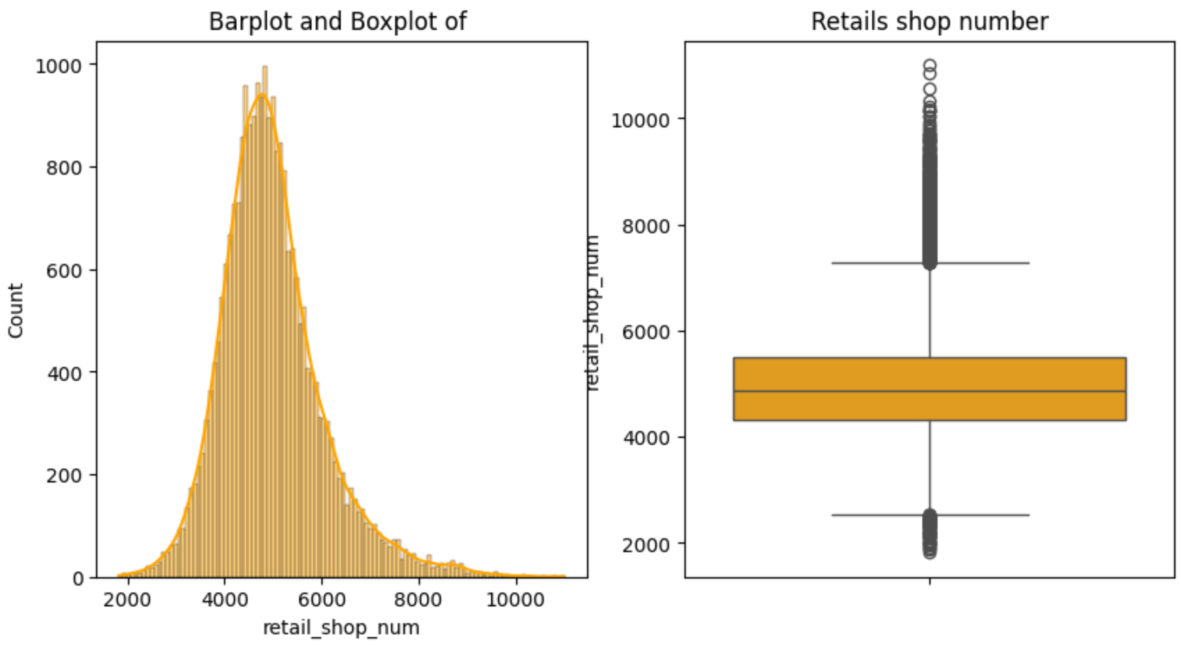
Values "Large" and "Mid" are populated more than compared to "Small". The company has more "Large" type of warehouses which is a good sign that its product is being sold a lot.



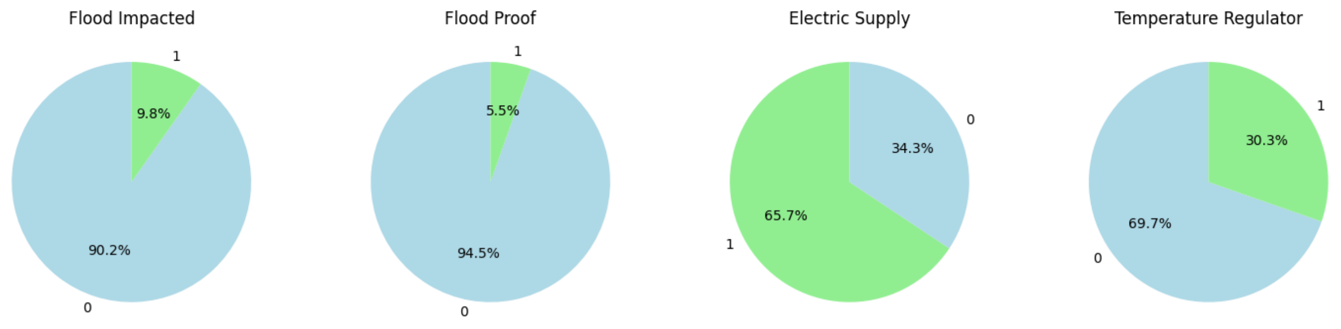
Refills required over three months – The number of refills has not crossed more than 8 over the course of three months. The frequency is almost close for all values except "2" which is the odd one. This means that a range of refills are being done consistently. We do have quite a few refills as "0" and "1" those warehouses effectively are low-income generators.



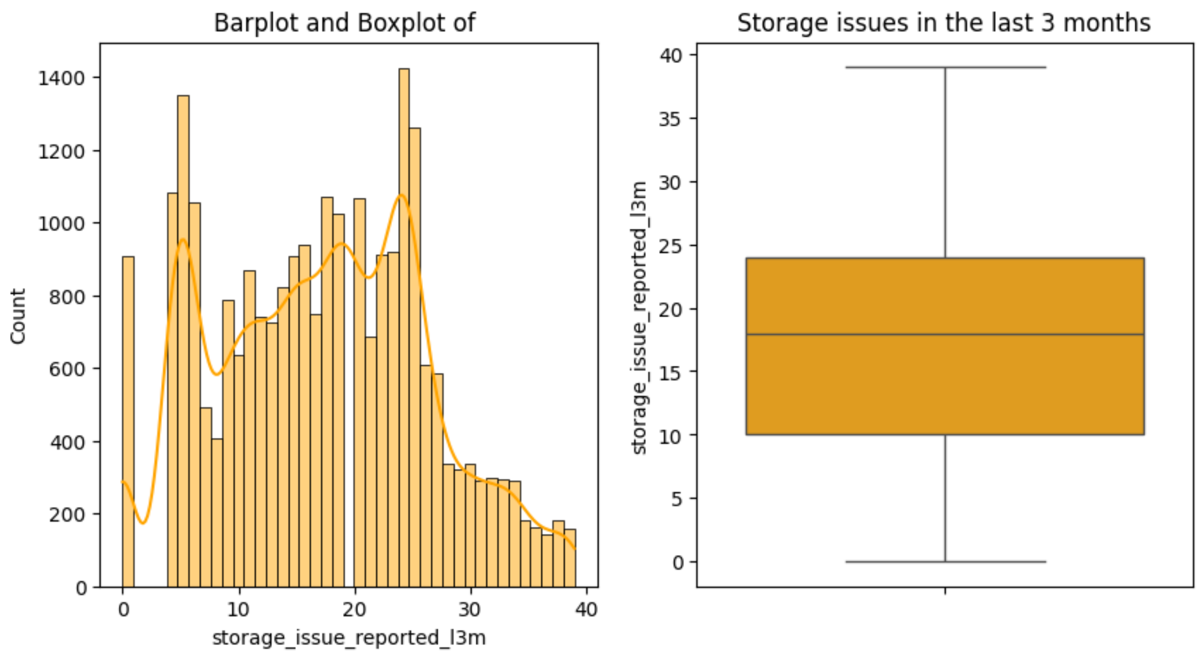
Number of Retail shops – The below count plot shows a close to normal yet slightly skewed to the right. The mean stands around 5000 shops, which is a very good number. There are some values that cross 10000, but because they are less, are considered as outliers. Outliers are present in both ends, these values do not need a fix, we must consider them truthful and prepare our models accordingly. The number of shops being in contact with the warehouses is a good indication of the company's ability to have a large consumer base.



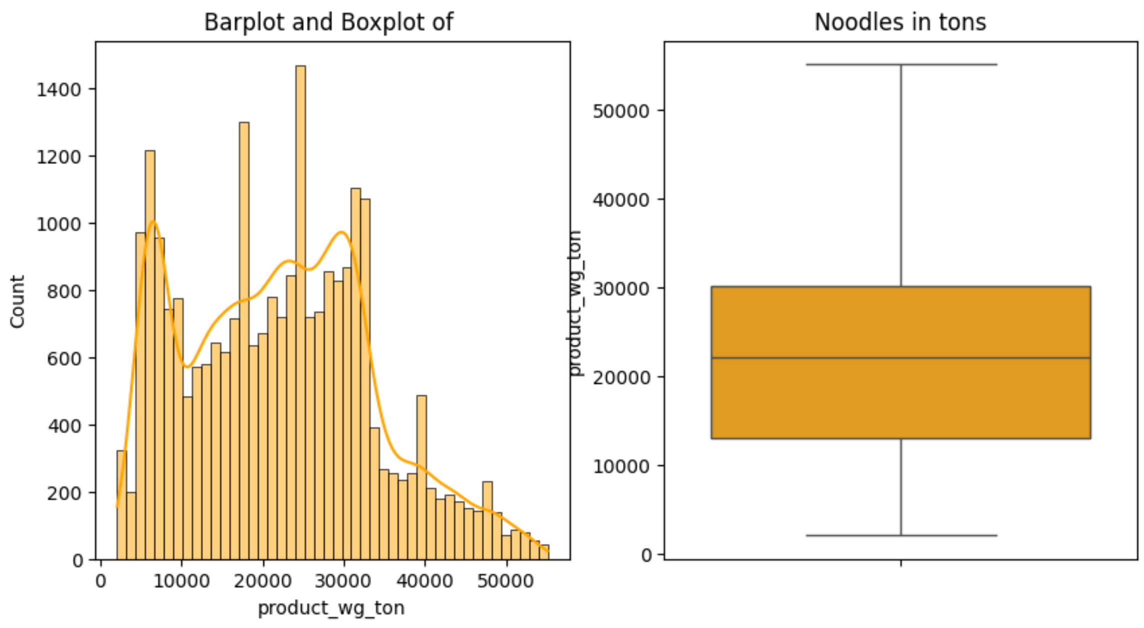
Most of the warehouses seem to be resilient to flooding, with 90.2% not affected by flood conditions. However, only a small fraction, 5.5%, are equipped to be flood proof. When it comes to utilities, 65.7% of the warehouses are supplied with electricity. The presence of temperature control systems is less common, with just 30.3% of warehouses having temperature regulators, leaving the majority without this feature.



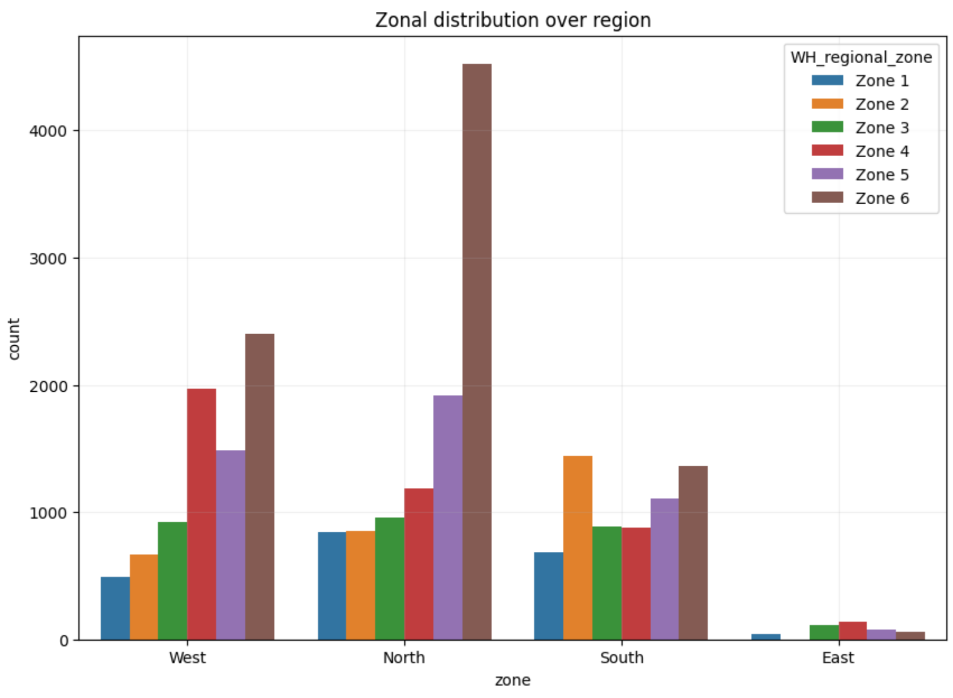
Storage issues in the last 3 months – The frequency is not evenly distributed; it is quite random. Being random is good in the sense that there might not have been repeating problems, but the opposite can also be true that the warehouses are always prone to issues (we do not know the cause and type, only the count is discussed here). Seeing the fact that we have no outliers is not good, as that would mean our average is on the higher side. All of this is for the past 3 months alone, which is a cause for concern.

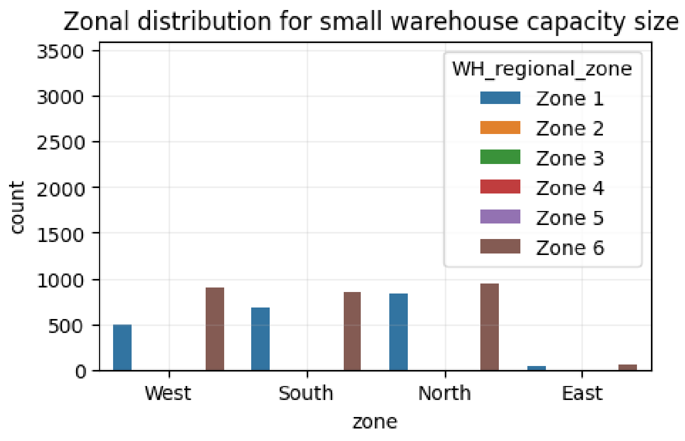


Product weight in tons – The product weight is our target feature, and it is closely like storage issues feature discussed above. The plot below explains the correlation between them in a better context and with more information. The only difference is we have a continuous stream of values unlike the other party which has a break.



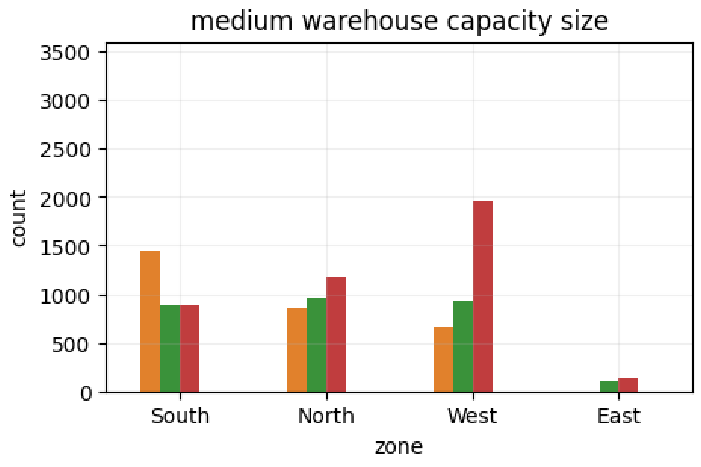
Zone and region – In the below plot we are looking at the count of warehouses segregated by directional and then again divided by regional zone. East has the least number of warehouses, irrespective of regional zone, moreover it has no zone 2 at all. Only zone 6 from northern region has a count of more than 4000 units, rest all are below 2500 only. The warehouse



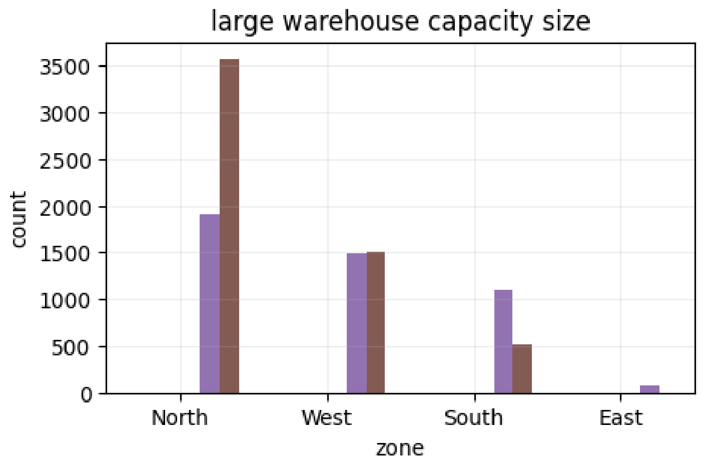


Zonal distribution on region-based company size gives us a new perspective on warehouse segregation.

When the warehouse capacity is "Small" then it is solely placed in zones 1 and 6 only. East region has the least count, which is expected to be, and north region has almost even distribution.



When the warehouse capacity is 'Mid" or medium then the distribution is again restricted to zones 2, 3, and 4. East as seen before does not have any warehouses in zone 2. Region west has the spotlight over north in this scenario.

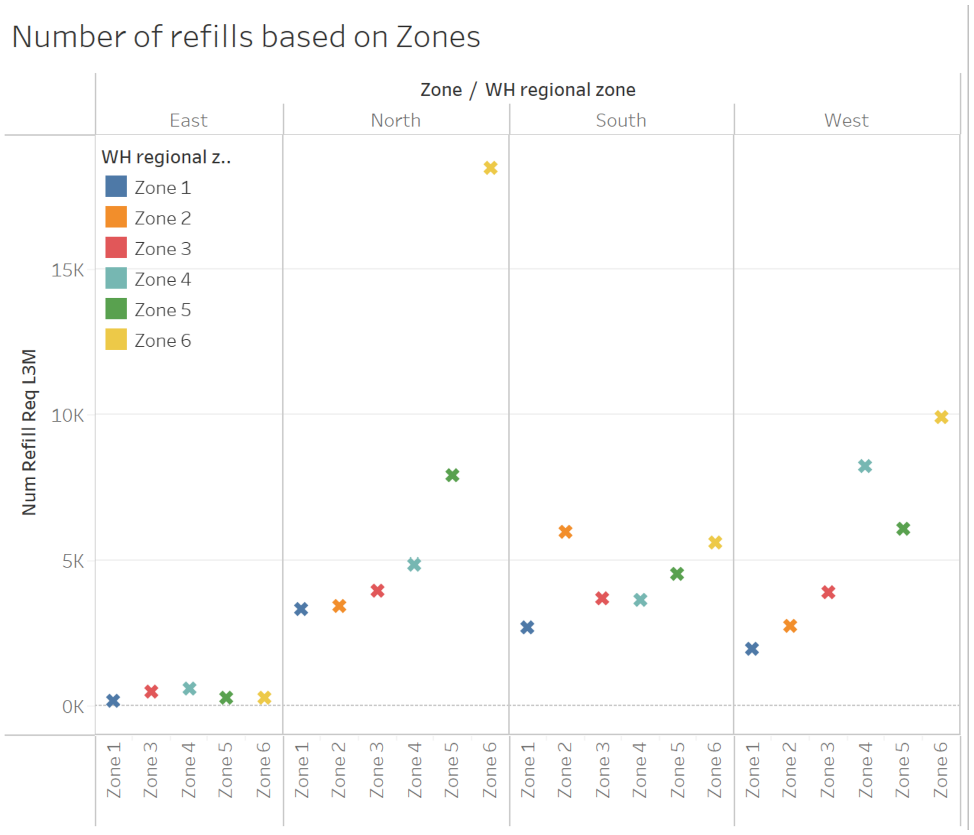


We can again see a similar pattern for "Large" company sizes. There seems to be some regional restriction. But the only tick here is that Zone 6 is repeated along with Zone 5. The combined count of small and large capacities has pushed zone 6 to have the maximum count of warehouses.

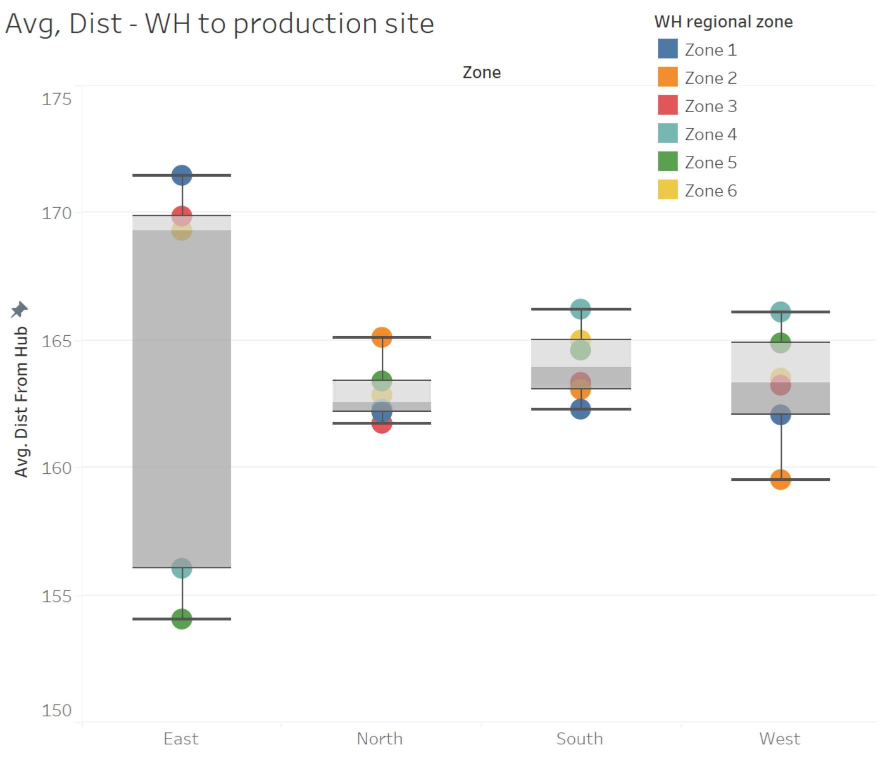
In "Large" warehouses, we don't see them being in zone 6. So, the east region has large warehouses only in zone 5.

The underlying reason for the warehouses being positioned as they are, is not known to us, but there seems to be compelling strategy in place.

__Refills done in the past 3 months segregated by zone –__ The below plot is very similar to the zonal distribution by region. This comes with no surprises as the number of warehouses increases in an area, the number of refills also seems to increase.



__Distances between warehouses and noodle production sites –__

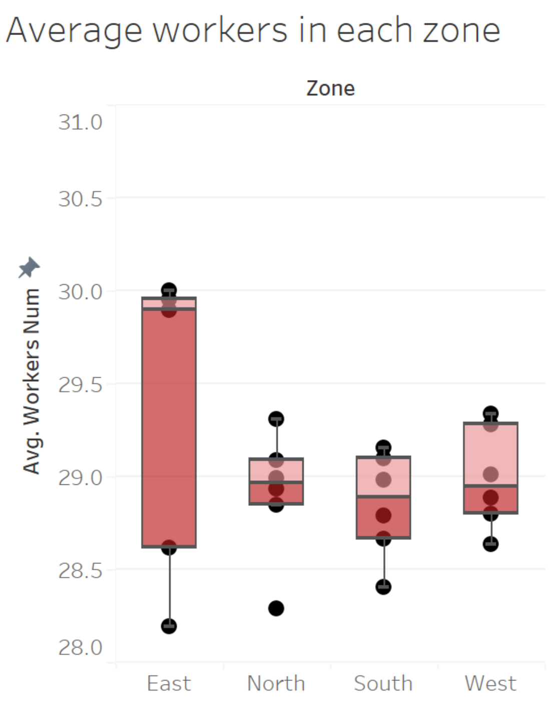


East while having the lowest set of warehouses in any zone contrastingly has the largest distance variation between the warehouses and productions sites. The quartiles alone are greater for it than outlier extremes for other regions.

North is exact opposite to east, with having the least distances between warehouses and production sites. The company's manufacturing facilities might be placed very near to the northern region.

On the right we see boxplot showing the average number of workers in each zone. Again, surprisingly we see a very contrasting take. East having the lowest number of warehouses still employs thirty works, while the others average out at 29 workers. The region of North keeps is very short and consistent.

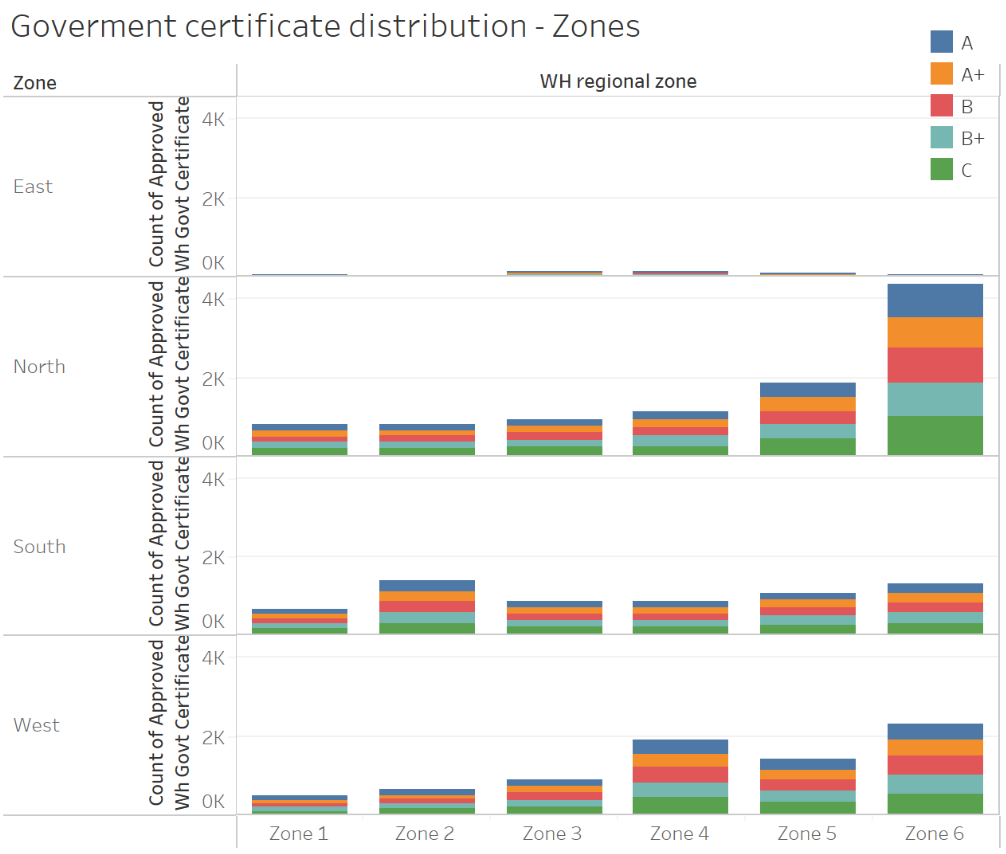
Eastern region shows the maximum variation while the others are consistent throughout. The northern region is also the only one to have an outlier. This shows how dense the values are.



The plot on the right shows the distribution of certificates provided by the government.

The segregation of the certificates from the looks of it seems consistent irrespective of the region. There are some variations on the zones, but they are not enough in magnitude to make a compelling difference.

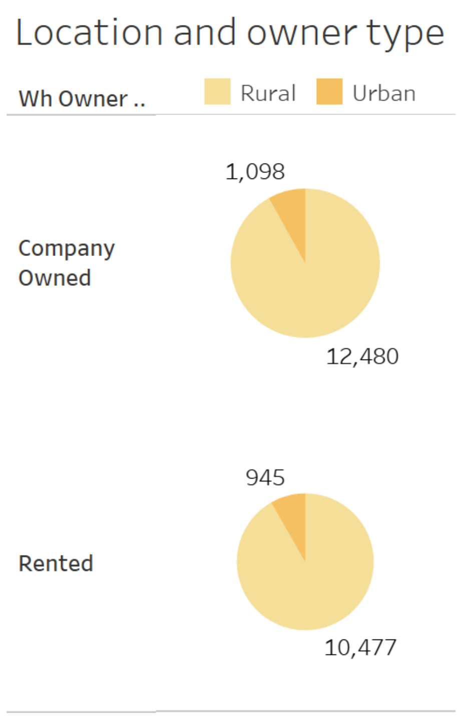
This plot might not provide us with much inference when building models.

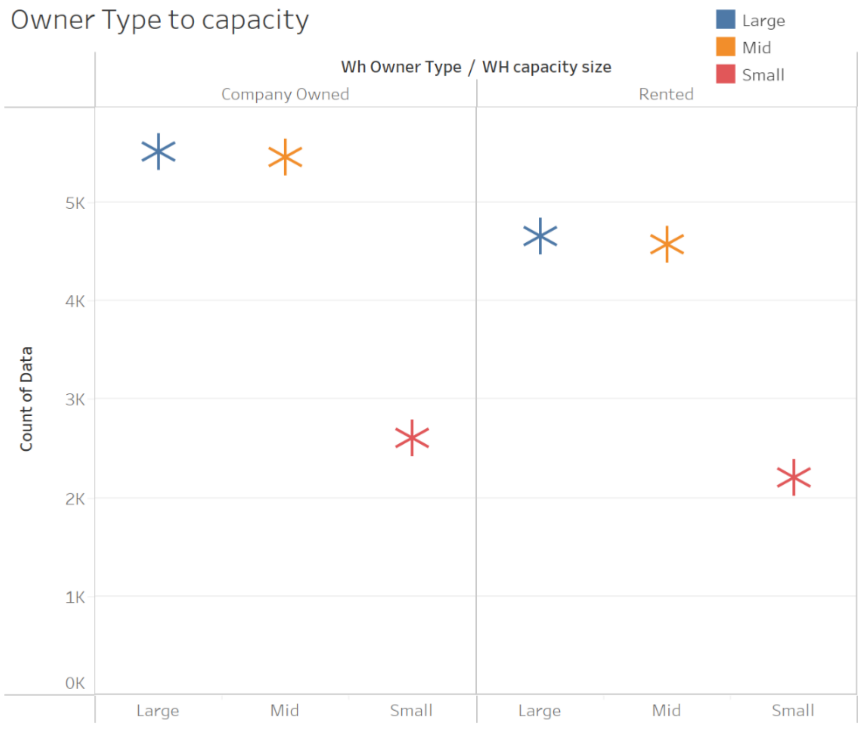


The data provided to us does not specify what each of the certificates represent or the said order, we can infer from the given values this order of precedence.

Order – A+ > A > B+ > B > C

Location & Owner type & Capacity – The below pie chart shows almost a consistent distribution between rural and urban locations and how they are owned. On an overall note, we can see that company owned warehouses are higher than rented but not by a huge margin. This is a good sign that the company is investing in its own warehouse locations. The plot on the right shows the owners capacity preferences. On all fronts company owned takes the lead, again a good sign. The difference is quite a bit.





Handling outliers – Outliers are not that common within the data, except a few like number of retails shops. Most of the features are numerical but are integer and are limited. Due to this we will let the outliers be.

Missing values – There are some missing values in the features "workers_num" and "approved_wh_govt_certificate", comparing the total number of missing values to total size of the data, there is a loss of "0.36%", This is a good thing as we can effectively remove them and still retain more than 99% of the original data. We will be removing those records that have null values, this is a huge boon as it keeps the data original without any human intervention making our models inclined much more to real world scenarios.

Insights from EDA –

- First and foremost, the data has an inherent record count imbalance for the feature “Location_Type”. The value “Rural” is enormously high in comparison to the other value “Urban”.
- When modelling the above issue will result in a model that works well for rural environments but the same cannot be said for the urban environments. The non-availability of precedence will hurt us.
- We must deal with this before we attempt to build models.
- Rest of the features are moderate in quality; directional zone is the other odd one where “East” is less in count to the other directions, but we can prepare artificial data to strengthen the model.
- We have quite a few features that numerical but are so constrained that they can be made into object for better model compatibility, of course we will be making the necessary processing based on the model we are going to build.
- “product_wg_ton” is our target feature and it is highly correlated to the “storage_issue_reported_l3m”, hence making sure this feature is processed optimally is important.

Clustering –

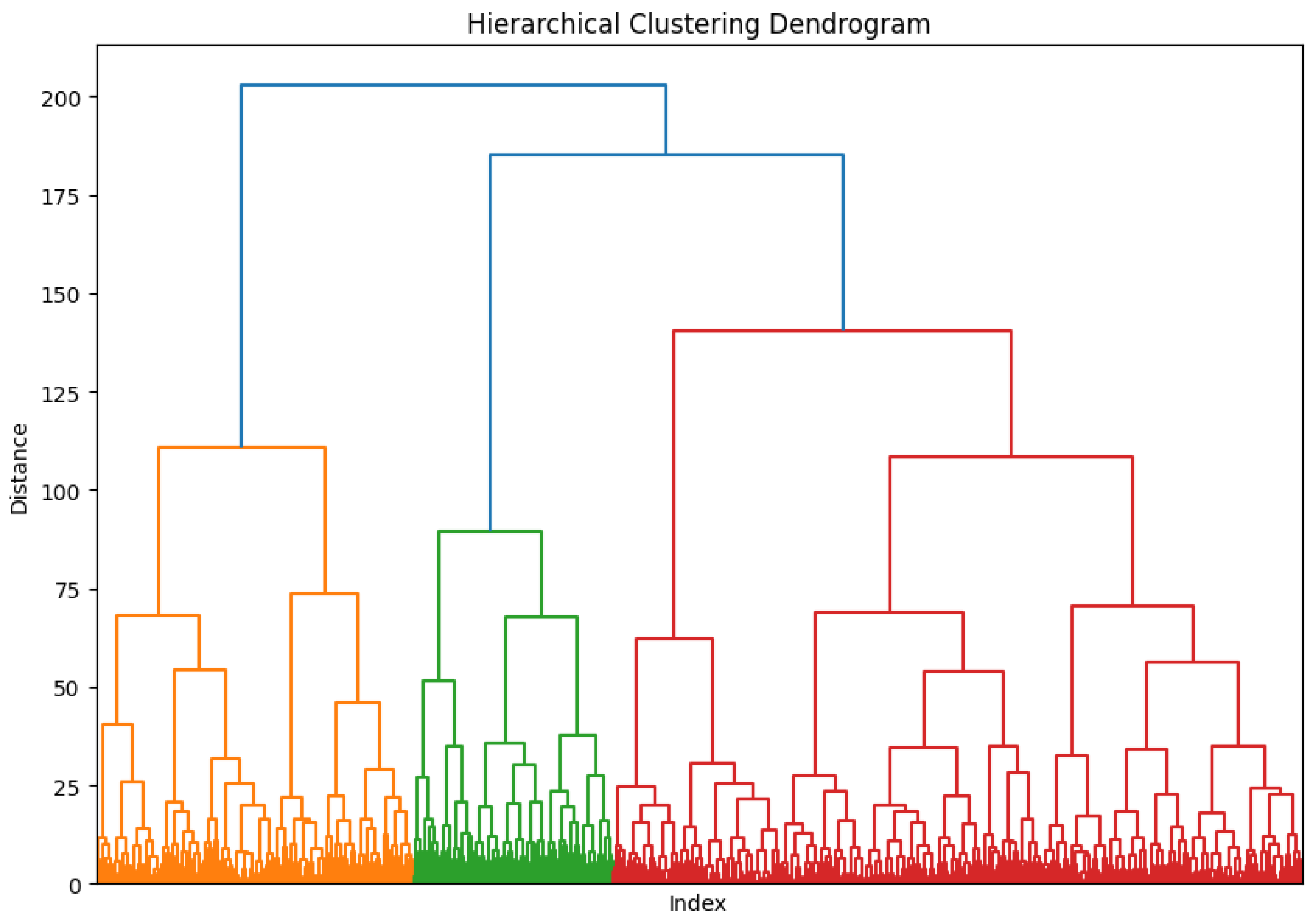
Clustering is a method to understand how the data is grouped. This is a method that we can use irrespective of the different categories given to us, to understand the hidden relationships.

For clustering it is not advisable to use the whole set of features that we have as they load the method with noise that makes it difficult to generate distinct clusters. Hence to find the most important features we have used "Recursive Feature Elimination (RFE)" to find the most important features.

Generating clustering models with the most important features we were able to find the best count of groups that best fit the data.

The cluster count was – "3".

The cluster count of "3" was chosen after determining metrics such as calinski_harabasz_score, davies_bouldin_score, and silhouette score.



What is the use of Clustering and how does that affect the direction of the problem –

The given problem statement states that – "Their higher management has notices that there is a miss match in the demand and supply. Where the demand is high, supply is pretty low and where the demand is low, supply is pretty high.".

The following is clear from the above statement – The target feature "product_wg_ton" is not a definitive answer which we can use to baseline but has now become a reference feature.

So now we are going to use the clusters that we have created and create a new target feature that would be used in-place of the current one. The new target feature would be the median value based on which cluster a particular instance of data belongs to. We then use classification.

A summary of what has been done to the data –

- Exploratory data analysis to find insights on data usage and structure.
- Treat null values using appropriate methods.
- Discuss about outliers and take optimal discourse.
- Perform clustering to understand the how the dataset can be clustered irrespective of the different categorical segregation we have.
- Create a new target feature encompassing the median value based on the cluster a particular row instance belongs.
- New target feature replaced the old target feature as there is an inherent value discrepancy and as such, they are not reliable metrics to build models.
- Encoding of categorical features to enable optimal model building.
- Create train and test datasets from the whole, in the ration of 67(train) to 33(test).

Decision Tree Model –

Approach -

Our strategy involved utilizing Grid Search, a hyperparameter optimization technique, to fine-tune our Decision Tree model which is renowned for its simplicity and effectiveness. This allowed us to identify the optimal settings for the model that ensures the data is adapted very well.

Model Performance -

The optimized Decision Tree model demonstrated exceptional performance and has set a high bar for a base line reference model.

Best Parameters: Max Depth = 10, Min Samples Leaf = 1, Min Samples Split = 2

Best Score: 99.12% → this is from grid-search.

Accuracy: 99.00%

These metrics indicate a model adept at capturing complex patterns without overfitting, and capable of making highly accurate decisions.

Precision, Recall, F1-Score, and Support -

Class Code	Precision	Recall	F1-Score	Support
18125.0	0.96	0.98	0.97	1259
23105.0	1.00	0.99	0.99	4359
25100.0	0.99	0.99	0.99	2017

The three classes after clustering are given above, they are the mean of each cluster that embodies the new target feature prepared. Precision measures the model's accuracy in predicting a class. A precision of 1.00 for class 23105.0, for example, indicates that every prediction made by the model for this class was correct. Recall assesses the model's ability to identify all relevant instances of a particular class. A recall of 0.99 for class 23105.0 suggests that the model identified 99% of all actual instances of this class. F1-Score is the harmonic mean of precision and recall, providing a single metric to assess the balance between them. High F1-scores across all classes, as seen in our model, denote a well-balanced performance between precision and recall, crucial for maintaining model reliability across diverse data points. Support, this metric indicates the number of actual occurrences of each class in the dataset. It's vital for understanding the distribution of classes and ensuring that the model's performance metrics are not skewed by imbalances in the data. Class "23105.0" does seem to pull in the most occurrences.

Decision Tree model stands as a very high baseline solution in our classification problem, exhibiting high accuracy, interpretability, and a solid foundation for future model comparisons.

Random Forest Classifier Model –

Approach –

Our methodology extended to leveraging the Random Forest Classifier, a more complex ensemble model known for its high accuracy and robustness against overfitting. The optimization process was carried out using Grid Search, allowing us to refine the model through the identification of optimal hyperparameters. This strategy ensured that the model could capture the intricate patterns within the data efficiently.

Model Performance –

The Random Forest Classifier emerged as a strong contender, exhibiting outstanding performance metrics that reinforce its reliability as a predictive model.

Best Parameters: Max Depth = 30, Min Samples Leaf = 1, Min Samples Split = 2, N_estimators = 200

Best Score: 98.45% – this score emanates from the grid-search optimization phase.

Accuracy: 98.38%

These indicators highlight the model's capability to handle complex patterns without succumbing to overfitting, ensuring accurate predictions across.

Precision, Recall, F1-Score, and Support -

Class Code	Precision	Recall	F1-Score	Support
18125.0	0.95	0.95	0.95	1259
23105.0	1.00	0.99	0.99	4359
25100.0	0.98	0.99	0.99	2017

The classification report reveals the model's proficiency in accurately predicting each class with high precision and recall, resulting in significant F1-scores. The Random Forest Classifier maintains high accuracy across all classes and is also well-balanced in terms of performance, ensuring reliability and robustness in multi-class classification.

The model's performance, particularly in class "23105.0", which has the highest occurrences, underscores its effectiveness in dealing with prevalent classes while maintaining high precision and recall rates. It seems to have slipped ever so slightly with the least occurred class of "18125.0" to a Precision and Recall score of 0.95, even the F1-Score is the same.

The Random Forest Classifier has proven to be a formidable model within our classification toolkit, standing out for its accuracy, ability to generalize across different data points, its performance is almost on par with our baseline decision tree. This demonstrates the potential of ensemble methods in achieving superior predictive accuracy and model robustness.

XGBoost (Extreme Gradient Boosting) Model –

Gradient Boosting –

Gradient Boosting is a powerful machine learning technique that builds models in a stage-wise fashion. XGBoost, or Extreme Gradient Boosting, enhances this approach through advanced regularization, parallel processing and enhanced flexibility in model tuning.

Approach –

In our exploration of advanced ensemble techniques, we adopted XGBoost for its efficiency, flexibility, and performance. Our optimization journey involved an exhaustive Grid Search over 108 candidate settings across 5 folds, totaling 540 fits (basically a lot). This meticulous process allowed us to identify the best combination of hyperparameters that tailor the model precisely to our dataset's characteristics.

Model Performance –

The XGBoost model has showcased exceptional performance, setting a new benchmark for accuracy and efficiency in our classification tasks.

Best Parameters: Colsample_bytree = 0.8, Learning Rate = 0.2, Max Depth = 8, N_estimators = 100, Subsample = 0.8

Best Score: 99.42% – indicating the model's superior fit during the grid-search phase.

Accuracy: 99.33%

These metrics underscore the model's ability to intricately understand and predict complex patterns, significantly reducing error rates compared to other models.

Precision, Recall, F1-Score, and Support -

Class Code	Precision	Recall	F1-Score	Support
0 (18125.0)	0.98	0.98	0.98	1259
1 (23105.0)	1.00	1.00	1.00	4359
2 (25100.0)	0.99	1.00	0.99	2017

XGBoost's performance in precision, recall, and F1-score across all classes reveals its exceptional capability in making accurate predictions, that nearly every prediction made for each class is correct. The model is robust and is also highly reliable. Class "1", with the highest support, showcases the model's strength in correctly identifying the most prevalent class without sacrificing accuracy on less common classes, a trait seen in the previous models as well.

The XGBoost model stands to be the best one for now.

K-Nearest Neighbours (KNN) Model –

KNN –

The K-Nearest Neighbour’s (KNN) algorithm is a simple, yet powerful machine learning technique used for classification and regression tasks. It classifies each data point based on the majority class among its 'k' nearest neighbours. The algorithm assumes that similar things exist in proximity.

Approach –

In exploring a variety of models, we also implemented the KNN algorithm, known for its straightforward implementation of data distributions. Our optimization process involved a Grid Search over combinations of hyperparameters, specifically focusing on the number of neighbours and the weighting method used in prediction. This process aimed to tailor the KNN model to best fit our dataset's unique characteristics.

Model Performance –

The KNN model, while not reaching the high benchmarks set by more complex models, still provided valuable insights into the dataset's structure.

Best Parameters: Metric = Manhattan, N_neighbors = 9, Weights = Distance

Best Score: 55.38% – reflecting the model's fit to the training data.

Accuracy: 55.42%

Although the performance metrics indicate a moderate understanding of the data, they highlight the challenges KNN faces in high-dimensional spaces or in the presence of categorical features.

Precision, Recall, F1-Score, and Support -

Class Code	Precision	Recall	F1-Score	Support
18125.0	0.31	0.07	0.12	1259
23105.0	0.59	0.87	0.70	4359
25100.0	0.38	0.18	0.24	2017

These metrics reveal the KNN model's limitations in achieving high precision and recall across the board, particularly for less prevalent classes. The model's best performance was observed in class “23105.0”, where it managed a precision of 0.59 and a recall of 0.87, leading to a relatively higher F1-score of 0.70. This suggests that while KNN can identify the dominant class which is greater than a coin toss probability. Its ability to distinguish among less common classes is largely constrained.

The KNN model's insights shows that the model is severely capped somewhere and is not able to perform better in comparison to other models. Its performance underscores the importance of selecting appropriate models based on the data's nature and the complexity of the classification problem at hand.

Naive Bayes Model –

Naive Bayes –

The Naive Bayes algorithm is a collection of classification algorithms based on Bayes' Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e., every pair of features being classified is independent of each other.

Approach –

In our pursuit to explore various models for our classification task, we applied the Naive Bayes algorithm, known for its effectiveness and efficiency, especially in text classification tasks. Through a focused Grid Search involving a minimal set of hyperparameters, we aimed to find the optimal smoothing parameter (alpha) for our model.

Model Performance –

The Naive Bayes model has demonstrated commendable performance, showcasing its utility in handling classification problems with notable accuracy.

Best Parameters: Alpha = 10.0

Best Score: 87.95% – indicating a high level of fit during the optimization phase.

Accuracy: 88.57%

These results reflect the model's adeptness at navigating the dataset, achieving substantial accuracy with a simple yet effective approach.

Precision, Recall, F1-Score, and Support -

Class Code	**Precision**	**Recall**	**F1-Score**	**Support**
18125.0	0.66	0.84	0.74	1259
23105.0	0.97	0.90	0.93	4359
25100.0	0.90	0.88	0.89	2017

These metrics illustrate the model’s strength in some respects, particularly notable in its precision for class “23105.0” and recall for class “18125.0”. The balanced F1-scores across classes affirm the model's efficacy in harmonizing precision and recall, showcasing its robustness in multi-class classification tasks to a good extent, but it is still not exemplary in performance.

The Naive Bayes model, with its good accuracy and insightful classification report, stands as an effective and efficient model in our classification toolkit but only for the class “23105.0”. Its performance underscores the value of incorporating probabilistic models to some extent.

Simple Gradient Boosting Model –

Gradient Boosting –

Gradient Boosting is a versatile machine learning algorithm that can be used for both regression and classification tasks. It builds models in a sequential manner, where each new model corrects errors made by previously trained models. Gradient Boosting combines multiple weak prediction models, typically decision trees, to create a strong predictive model.

Approach –

Our exploration of ensemble methods included the Simple Gradient Boosting model, known for its precision and adaptability. The model was optimized through a comprehensive Grid Search process, which tested a variety of hyperparameters across multiple folds. This strategy was crucial in pinpointing the best combination that maximizes the model’s performance on our dataset.

Model Performance –

The Simple Gradient Boosting model has emerged as a top performer, with metrics that highlight its superior predictive capabilities.

Best Parameters: Learning Rate = 0.2, Max Depth = 5, N_estimators = 200

Best Score: 99.44% – reflecting an exceptional fit to the training data.

Accuracy: 99.45%

These results underscore the model's proficiency in understanding and predicting the dataset's patterns, outperforming several benchmarks with its high accuracy and efficiency.

Precision, Recall, F1-Score, and Support -

Class Code	Precision	Recall	F1-Score	Support
18125.0	0.98	0.98	0.98	1259
23105.0	1.00	1.00	1.00	4359
25100.0	0.99	1.00	1.00	2017

The classification report for the Gradient Boosting model illustrates its remarkable precision and recall, leading to perfect or near-perfect F1-scores across all classes. This level of performance signifies the model's capability to accurately identify and classify each category, even in a multi-class setting, making it a robust and reliable choice for our classification tasks.

The Simple Gradient Boosting model, with its impressive accuracy and detailed classification insights, stands as a compelling solution in our suite of classification models.

SVM (Support Vector Machine) Model –

SVM –

Support Vector Machines (SVM) are a set of supervised learning methods used for classification, regression, and outliers' detection. The core idea of SVM is to find the optimal hyperplane (in two-dimensional space, a line) that best separates different classes by maximizing the margin between the closest points of the classes, which are called support vectors. SVMs are particularly well-suited for complex classification problems with high-dimensional spaces, where the number of dimensions exceeds the number of samples. Its capability to use different kernel functions to transform the input space into a higher-dimensional space makes it a versatile and powerful classification algorithm.

Approach –

In our exploration of various classification models, we applied the SVM algorithm to leverage its high-dimensional classification capabilities. Utilizing Grid Search to navigate through a range of hyperparameters, we identified the optimal settings that maximize the model's performance on our dataset.

Model Performance - The SVM model demonstrated strong performance, validating its efficacy in high-dimensional classification tasks.

Best Parameters: C = 1, Gamma = 'scale', Kernel = 'linear'

Best Score: 96.53% -- result of a grid-search.

Accuracy: 96.27%

These results underscore the SVM model's adeptness at accurately classifying data, benefitting from the linear kernel's simplicity and the regularization parameter's balance between model complexity and classification accuracy.

Precision, Recall, F1-Score, and Support -

Class Code	Precision	Recall	F1-Score	Support
18125.0	0.87	0.90	0.89	1259
23105.0	0.98	0.98	0.98	4359
25100.0	0.97	0.97	0.97	2017

The table above reveals the SVM model's high precision and recall across all classes, with impressive F1-scores that highlight its balanced performance. Particularly, its ability to distinguish between the various classes with high accuracy suggests it strength in understanding the data that we have given to it. It did slip quite a bit in terms of the least occurring class, but overall, the performance has been commendable, and it is very close to our baseline decision tree.

The SVM model stands as a robust tool in our classification arsenal, offering a high degree of accuracy and efficiency.

Model Comparison –

Decision Trees offered a strong baseline with very high accuracy, allowing for easy interpretation and rapid predictions. Its performance was near-perfect, highlighting its capability in handling the data with minimal complexity.

Random Forest improved upon Decision Trees by leveraging an ensemble approach, enhancing accuracy and robustness against overfitting. It demonstrated high precision and recall, making it effective for complex classification problems.

XGBoost stood out for its advanced optimization and regularization capabilities, achieving superior accuracy. Its fine-tuning through Grid Search highlighted its flexibility and power in handling diverse datasets, making it a top performer. It was the second best model that we have with us for the problem statement.

K-Nearest Neighbors (KNN), while simpler, showed limitations in handling the dataset, reflecting lower accuracy and effectiveness. Its performance underscored the importance of considering model complexity and the dimensionality of the data. The model performance was comparable to a blind selection if not less.

Naive Bayes provided valuable insights with its probabilistic approach, offering good accuracy with computational efficiency. It proved particularly useful for baseline comparisons and scenarios where the probabilistic interpretation of classification is a must.

Simple Gradient Boosting demonstrated excellent performance, benefiting from its sequential learning approach to minimize errors progressively. Its high accuracy and balanced precision and recall across classes marked it as highly effective. The best model in our portfolio.

Support Vector Machine (SVM) showcased strong capabilities in high-dimensional spaces, with the linear kernel providing a solid foundation for accurate classifications. Its performance emphasized the SVM's utility in linearly separable data or when a linear approximation is sufficient.

Comparative Table –

Here's a table summarizing the key metrics of each model discussed:

Model	Best Score (Grid Search)	Test Accuracy	Precision, Recall, F1-Score Highlights
Decision Trees	99.12%	99.00%	High across all classes
Random Forest	98.45%	98.38%	Very high, especially in precision
XGBoost	99.42%	99.33%	Superior performance with nearly perfect scores
K-Nearest Neighbors	55.38%	55.42%	Lower performance, especially in precision and recall
Naive Bayes	87.95%	88.57%	Good, with high efficiency and computational speed
Simple Gradient Boosting	99.44%	99.45%	Excellent precision and recall, balanced F1-scores
SVM	96.53%	96.27%	Strong in high-dimensional spaces, linear separability

What can we conclude –

Based on the comparative analysis, the **XGBoost** and **Simple Gradient Boosting** models showcased the highest performance metrics, both in terms of Best Score from Grid Search and Test Accuracy. Among these, **Simple Gradient Boosting** slightly edged out with its simpler mechanics of optimization, making it the best model for our discussion. Its ability to handle complex datasets such as what we gave as input, with high precision and recall across multiple classes makes it a robust choice for addressing classification challenges.

Implementing an **SGB model** in a manufacturing facility dealing with inventory distribution challenges to warehouses can lead to significant operational improvements. By ensuring that the right amount of product is sent to the right place at the right time, the model not only addresses current inefficiencies but also lays the groundwork for a more responsive and agile supply chain.

We can also implement **XGB as an alternative to SGB as the performance delta between them is very minimal. On the whole decision trees seemed to be the best fit for the data frame that we have.**

Recommendations

- **In-depth Demand-Supply Review**: To conduct a granular analysis of the current demand-supply matching process to identify why the gaps are present. Understanding the root cause of these imbalances is crucial for long-term optimization.
- **Geographical Distribution**: Investigate the disparity in warehouse locations, focusing on the rural-urban distribution. Tailoring our supply chain strategy to address these geographical imbalances will help in better serving all market segments.
- **Quality Improvement in data**: To address the high volume of missing or data points in our dataset. Ensuring data integrity will significantly enhance the accuracy of our predictive models and our inventory management capabilities.
- **Adopting the Simple Gradient Boosting Model**: Utilize the SGB model for dynamic forecasting of warehouse stock levels. Its superior performance and accuracy make it an ideal tool for real-time inventory optimization. We can also employ the XGB model as they are mostly similar not only in performance but also in inner workings.
- **Real-Time Data Analytics**: Explore the integration of real-time data analytics into our supply chain management. This will allow for more agile responses to market changes and demand fluctuations.
- **Targeted Marketing Strategies**: Leverage insights into demand patterns to drive targeted marketing efforts, especially in high-demand regions. This approach will not only increase sales but also enhance customer satisfaction through better product availability.
- What was done now is not the final setup but the first in a series of many steps that we must go through to once this has been used.
- Using the current supply as base we can follow up with new metrics, understand and optimize to negate understocking and overstocking.
- After implementing this we should keep track on how the warehouses react to a shipment, we again collect the info on a granular scale for further modelling.